Chapter 5

Constructing Learning Algorithms

To implement the SRM inductive principle in learning algorithms one has to minimize the risk in a given set of functions by controlling two factors: the value of the empirical risk and the value of the confidence interval.

Developing such methods is the goal of the theory for constructing learning algorithms.

In this chapter we describe learning algorithms for pattern recognition and consider their generalizations for the regression estimation problem.

5.1 WHY CAN LEARNING MACHINES GENERALIZE?

The generalization ability of learning machines is based on the factors described in the theory for controlling the generalization ability of learning processes. According to this theory, to guarantee a high level of generalization ability of the learning process one has to construct a structure

$$S_1 \subset S_2 \subset, \ldots, \subset S$$

on the set of loss functions $S = \{Q(z, \alpha), \alpha \in \Lambda\}$ and then choose both an appropriate element S_k of the structure and a function $Q(z, \alpha_\ell^k) \in S_k$ in this element that minimizes the corresponding bounds, for example, bound (4.1). The bound (4.1) can be rewritten in the simple form

$$R(\alpha_\ell^k) \leq R_{\text{emp}}(\alpha_\ell^k) + \Phi(\frac{\ell}{h_k}), \tag{5.1}$$

where the first term is the empirical risk and the second term is the confidence interval.

There are two *constructive* approaches to minimizing the right-hand side of inequality (5.1).

In the first approach, during the design of the learning machine one determines a set of admissible functions with some VC dimension h^*. For a given amount ℓ of training data, the value h^* determines the confidence interval $\Phi(\frac{\ell}{h^*})$ for the machine. Choosing an appropriate element of the structure is therefore a problem of designing the machine for a specific amount of data.

During the learning process this machine minimizes the first term of the bound (5.1) (the number of errors on the training set).

If for a given amount of training data one designs too complex a machine, the confidence interval $\Phi(\frac{\ell}{h^*})$ will be large. In this case even if one could minimize the empirical risk down to zero the number of errors on the test set could still be large. This phenomenon is called *overfitting*.

To avoid overfitting (to get a small confidence interval) one has to construct machines with small VC dimension. On the other hand, if the set of functions has a small VC dimension, then it is difficult to approximate the training data (to get a small value for the first term in inequality (5.1)). To obtain a small approximation error and simultaneously keep a small confidence interval one has to choose the architecture of the machine to reflect *a priori* knowledge about the problem at hand.

Thus, to solve the problem at hand by these types of machines, one first has to find the appropriate architecture of the learning machine (which is a result of the trade off between overfitting and poor approximation) and second, find in this machine the function that minimizes the number of errors on the training data. This approach to minimizing the right-hand side of inequality (5.1) can be described as follows:

Keep the confidence interval fixed (by choosing an appropriate construction of machine) and minimize the empirical risk.

The second approach to the problem of minimizing the right-hand side of inequality (5.1) can be described as follows:

Keep the value of the empirical risk fixed (say equal to zero) and minimize the confidence interval.

Below we consider two different types of learning machines that implement these two approaches:

(i) Neural Networks (which implement the first approach), and

(ii) Support Vector machines (which implement the second approach).

Both types of learning machines are generalizations of the learning machines with a set of linear indicator functions constructed in the 1960s.

5.2 SIGMOID APPROXIMATION OF INDICATOR FUNCTIONS

Consider the problem of minimizing the empirical risk on the set of *linear indicator functions*

$$f(x, w) = \text{sign}\{(w \cdot x)\}, \quad w \in R^n, \tag{5.2}$$

where $(w \cdot x)$ denotes a inner product between vectors w and x. Let

$$(x_1, y_1), \ldots, (x_\ell, y_\ell)$$

be a training set, where x_j is a vector, and $y_j \in \{1, -1\}$, $j = 1, \ldots, \ell$.

The goal is to find the vector of parameters w_0 (weights) which minimize the empirical risk functional

$$R_{\text{emp}}(w) = \frac{1}{\ell} \sum_{j=1}^{\ell} (y_j - f(x_j, w))^2. \tag{5.3}$$

If the training set is separable without error (i.e. the empirical risk can become zero) then there exists a finite step procedure that allows us to find such a vector w_0, for example the procedure that Rosenblatt proposed for the perceptron (see the Introduction).

The problem arises when the training set cannot be separated without errors. In this case the problem of separating the training data with the smallest number of errors is NP-complete. Moreover, one cannot apply regular gradient based procedures to find a local minimum of functional (5.3), since for this functional the gradient is either equal to zero or undefined.

Therefore, the idea was proposed to approximate the indicator functions (5.2) by so-called *sigmoid functions* (see Fig. 0.3)

$$\bar{f}(x, w) = S\{(w \cdot x)\}, \tag{5.4}$$

where $S(u)$ is a smooth monotonic function such that

$$S(-\infty) = -1, \; S(+\infty) = 1,$$

for example,

$$S(u) = \tanh u = \frac{\exp(u) - \exp(-u)}{\exp(u) + \exp(-u)}.$$

For the set of sigmoid functions, the empirical risk functional

$$R_{\text{emp}}(w) = \frac{1}{\ell} \sum_{j=1}^{\ell} (y_j - S\{(w \cdot x)\})^2$$

is smooth in w. It has a gradient

$$\text{grad}_w R_{\text{emp}}(w) = -\frac{2}{\ell}\sum_{j=1}^{\ell}[y_j - S((w \cdot x_j))]\, S'\{(w \cdot x_j)\}x_j^T$$

and therefore it can be minimized using standard gradient-based methods, for example, the *gradient descent method*:

$$w_{\text{new}} = w_{\text{old}} - \gamma(\cdot)\text{grad}\, R_{\text{emp}}(w_{\text{old}}),$$

where $\gamma(\cdot) = \gamma(n) \geq 0$ is a value which depends on the iteration number n. For convergence of the gradient descent method to local minima it is sufficient that the values of gradient are bounded and that coefficients $\gamma(n)$ satisfies the following conditions:

$$\sum_{n=1}^{\infty}\gamma(n) = \infty, \qquad \sum_{n=1}^{\infty}\gamma^2(n) < \infty.$$

Thus, the idea is to use the sigmoid approximation at the stage of estimating the coefficients, and use the threshold functions (with the obtained coefficients) for the last neuron at the stage of recognition.

5.3 NEURAL NETWORKS

In this section we consider classical neural networks, which implement the first strategy: keep the confidence interval fixed and minimize the empirical risk.

This idea is used to estimate the weights of all neurons of a multi-layer perceptron (Neural Network). Instead of linear indicator functions (single neurons) in the networks one considers a set of sigmoid functions.

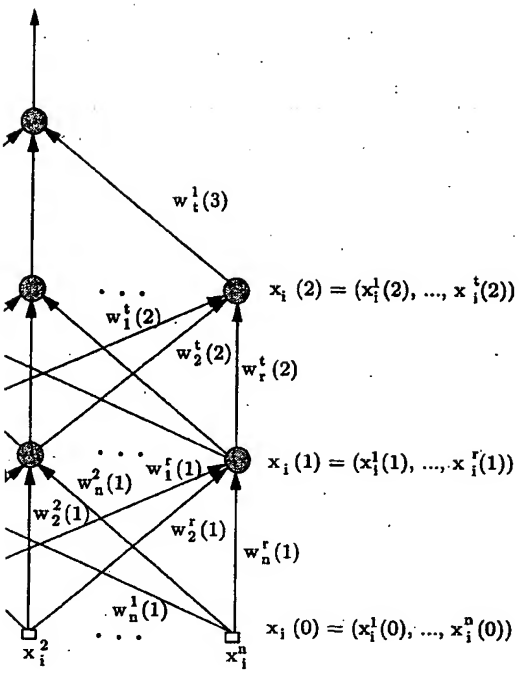
The method for calculating the gradient of the empirical risk for the sigmoid approximation of neural networks, called the *back-propagation method* was proposed[1] in 1986 (Rumelhart, Hinton, and Williams, 1986), (LeCun, 1986).

Using this gradient, one can iteratively modify the coefficients (weights) of a neural net on the basis of standard gradient-based procedures.

5.3.1 The Back-Propagation Method

To describe the back-propagation method we use the following notations (Fig. 5.1):

[1] See footnote 5 on page 12.



Network is a combination of several levels of sigmoid one layer form the input for the next layer.

ntains $m+1$ layers: the first layer $x(0)$ describes $= (x^1, ..., x^n)$. We denote the input vector by

$_i = (x_i^1(0), ...x_i^n(0)), \quad i = 1, ..., \ell,$

he input vector $x_i(0)$ on the kth layer by

$i) = (x_i^1(k), ..., x_i^{n_k}(k)), \quad i = 1, ..., \ell,$

by n_k the dimensionality of the vectors $x_i(k)$ $i = ..., m-1$ can be any number, but $n_m = 1$).

onnected with the layer k through the $(n_k \times n_{k-1})$

$)x_i(k-1)\}, \quad k = 1, 2, \ldots, m, \quad i = 1, ..., \ell, \qquad (5.5)$

$-1)\}$ defines the sigmoid function of the vector

$= w(k)x_i(k-1) = (u_i^1(k), ..., u_i^{n_k}(k))$

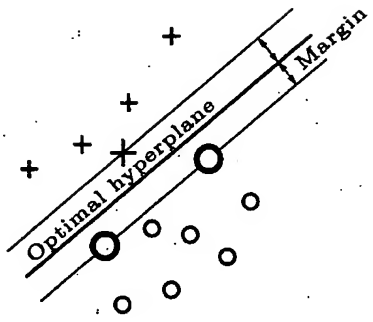


FIGURE 5.2. The Optimal separating hyperplane is the one that separates the data with maximal margin.

5.4.2 The Structure of Canonical Hyperplanes

Now let separating hyperplanes is defined on the set of vectors

$$X^* = x_1, \ldots, x_r,$$

bounded by a sphere of the radius R

$$|x_i - a| \leq R, \quad x_i \in X^*$$

(a is the center of the sphere). Consider a set of hyperplanes in *canonical form* (with respect to these vectors) defined by the pairs (w, b) satisfying the condition

$$\min_{x_i \in X^*} |(w \cdot x_i) + b| = 1.$$

Note that the set of canonical separating hyperplanes coincides with the set of all separating hyperplanes. It only specifies the normalization of the parameters of hyperplanes.

The idea of constructing a machine that fixes the empirical risk and minimizes the confidence interval is based on the existence of the following bound on the VC dimension of canonical hyperplanes.

Theorem 5.1. *A subset of canonical hyperplanes*

$$f(x, w, b) = sign\{(w \cdot x) + b\},$$

defined on X^* and satisfying the constraint

$$||w|| \leq A$$

has the VC dimension h bounded by the inequality

$$h \leq min\left([R^2 A^2], n\right) + 1.$$

In Section 3.5 we stated that the VC dimension of the set of hyperplanes is equal to $n+1$, where n is dimensionality of the space. However, the VC dimension of the *subset* of the set of hyperplanes, with canonical form satisfying $|w|^2 \leq A^2$, can be less.[2]

Below we consider hyperplanes only in canonical form, constructed on the basis of the training vectors $X^* \triangleq x_1, ..., x_\ell$.[3] For simplicity we call them hyperplanes.

Let us construct the structure on the set of hyperplanes by increasing the norm of the weights w. Then in order to obtain the smallest probability of error on the test set, we choose the hyperplane from the element of the structure which separates the training data and whose element of the structure gives the smallest bound on the VC dimension, that is, with the smallest norm of weights.

5.5 CONSTRUCTING THE OPTIMAL HYPERPLANE

To construct the Optimal hyperplane one has to separate the vectors x_i of the training set

$$(y_1, x_1), \ldots, (y_\ell, x_\ell)$$

belonging to two different classes $y \in \{-1, 1\}$ using the hyperplane with the smallest norm of coefficients.

To find this hyperplane one has to solve the following quadratic programming problem: minimize the functional

$$\Phi(w) = \frac{1}{2}(w \cdot w) \tag{5.10}$$

under the constraints of inequality type

$$y_i[(x_i \cdot w) + b] \geq 1, \quad i = 1, 2, \ldots, \ell. \tag{5.11}$$

The solution to this optimization problem is given by the saddle point of the Lagrange functional (Lagrangian):

$$L(w, b, \alpha) = \frac{1}{2}(w \cdot w) - \sum_{i=1}^{\ell} \alpha_i \{[(x_i \cdot w) + b] y_i - 1\}, \tag{5.12}$$

where the α_i are Lagrange multipliers. The Lagrangian has to be minimized with respect to w, b and maximized with respect to $\alpha_i > 0$.

---

[2] In Section 5.7 we describe a separating hyperplane in 10^{13} dimensional space with relatively small estimate of the VC dimension ($\approx 10^3$).

[3] In Section 5.11 we will discuss this choice of the set X^*.

In the saddle point, the solutions w_0, b_0, and α^0 should satisfy the conditions

$$\frac{\partial L(w_0, b_0, \alpha^0)}{\partial b} = 0,$$

$$\frac{\partial L(w_0, b_0, \alpha^0)}{\partial w} = 0.$$

Rewriting these equations in explicit form one obtains the following properties of the Optimal hyperplane:

(i) The coefficients α_i^0 for the Optimal hyperplane should satisfy the constraints

$$\sum_{i=1}^{\ell} \alpha_i^0 y_i = 0, \quad \alpha_i^0 \geq 0, \quad i = 1, \ldots, \ell \tag{5.13}$$

(first equation).

(ii) The Optimal hyperplane (vector w_0) is a linear combination of the vectors of the training set.

$$w_0 = \sum_{i=1}^{\ell} y_i \alpha_i^0 x_i, \quad \alpha_i^0 \geq 0, \quad i = 1, \ldots, \ell \tag{5.14}$$

(second equation).

(iii) Moreover, only the so-called *support vectors* can have nonzero coefficients α_i^0 in the expansion of w_0. The support vectors are the vectors for which, in inequality (5.11), the equality is achieved. Therefore we obtain

$$w_0 = \sum_{\text{support vectors}} y_i \alpha_i^0 x_i, \quad \alpha_i^0 \geq 0. \tag{5.15}$$

This fact follows from the classical Kühn–Tucker theorem, according to which the necessary and sufficient conditions for the Optimal hyperplane are that the separating hyperplane satisfy the conditions:

$$\alpha_i^0 \{[(x_i \cdot w_0) + b_0] y_i - 1\} = 0, \quad i = 1, \ldots, \ell. \tag{5.16}$$

Putting the expression for w_0 into the Lagrangian and taking into account the Kühn–Tucker conditions, one obtains the functional

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j}^{\ell} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j). \tag{5.17}$$

It remains to maximize this functional in the non-negative quadrant

$$\alpha_i \geq 0, \quad i = 1, ..., \ell \tag{5.18}$$

under the constraint

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0. \tag{5.19}$$

According to Eq. (5.15) the Lagrange multipliers and support vectors determine the Optimal hyperplane. Thus, to construct the Optimal hyperplane one has to solve a simple quadratic programming problem: maximize the quadratic form (5.17) under constraints[4] (5.18) and (5.19).

Let $\alpha_0 = (\alpha_1^0, \ldots, \alpha_\ell^0)$ be a solution to this quadratic optimization problem. Then the norm of the vector w_0 corresponding to the Optimal hyperplane equals:

$$|w_0|^2 = 2W(\alpha_0) = \sum_{\text{support vectors}} \alpha_i^0 \alpha_j^0 (x_i \cdot x_j) y_i y_j.$$

The separating rule, based on the Optimal hyperplane, is the following indicator function

$$f(x) = \text{sign}\left(\sum_{\text{support vectors}} y_i \alpha_i^0 (x_i \cdot x) - b_0 \right), \tag{5.20}$$

where x_i are the support vectors, α_i^0 are the corresponding Lagrange coefficients, and b_0 is the constant (threshold)

$$b_0 = \frac{1}{2}\left[(w_0 \cdot x^*(1)) + (w_0 \cdot x^*(-1)) \right],$$

where we denote by $x^*(1)$ some (any) support vector belonging to the first class and we denote by $x^*(-1)$ a support vector belonging to the second class (Vapnik and Chervonenkis, 1974), (Vapnik, 1979).

5.5.1 *Generalization for the Nonseparable Case*

To construct the Optimal type hyperplane in the case when the data are linearly nonseparable, we introduce non-negative variables $\xi_i \geq 0$ and a

[4]This quadratic programming problem is simple because it has simple constraints. For the solution of this problem, one can use special methods which are fast and applicable for the case with a large number of support vectors ($\approx 10^4$ support vectors) (More and Toraldo, 1991). Note that in the training data the support vectors constitute only a small part of the training vectors (in our experiments 3% to 5%).

function

$$F_\sigma(\xi) = \sum_{i=1}^{\ell} \xi_i^\sigma$$

with parameter $\sigma > 0$.

Let us minimize the functional $F_\sigma(\xi)$ subject to constraints

$$y_i((w \cdot x_i) + b) \geq 1 - \xi_i, \qquad i = 1, 2, \ldots, \ell, \tag{5.21}$$

and one more constraint

$$(w \cdot w) \leq c_n. \tag{5.22}$$

For sufficiently small $\sigma > 0$ the solution to this optimization problem defines a hyperplane that minimizes number of training errors under condition that the parameters of this hyperplane belong to the subset (5.22) (to the element of the structure

$$S_n = \{(w \cdot x) + b : \ (w \cdot w) \leq c_n\}$$

determined by constant c_n).

For computational reasons, however, we consider the case $\sigma = 1$. This case corresponds to the smallest $\sigma > 0$ that is still computationally simple. We call this hyperplane the Generalized Optimal hyperplane.

1. One can show (using the technique described above) that the Generalized Optimal hyperplane is determined by the vector

$$w = \frac{1}{C^*} \sum_{i=1}^{\ell} \alpha_i y_i x_i,$$

where parameters α_i, $i = 1, \ldots, \ell$ and C^* are the solutions to the following convex optimization problem:

Maximize the functional

$$W(\alpha, C^*) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2C^*} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \frac{c_n C^*}{2}$$

subject to constraints

$$\sum_{i=1}^{\ell} y_i \alpha_i = 0$$

$$0 \leq \alpha_i \leq 1, \quad i = 1, \ldots, \ell$$

$$C^* \geq 0$$

2. To simplify computations one can introduce the following (slightly modified) concept of the Generalized Optimal hyperplane (Cortes and Vapnik, 1995). The Generalized Optimal hyperplane is determined by the vector w that minimizes the functional

$$\Phi(w,\xi) = \frac{1}{2}(w \cdot w) + C\left(\sum_{i=1}^{\ell} \xi_i\right)$$

(here C is a *given* value) subject to constraint (5.21).

The technique of solution of this quadratic optimization problem is almost equivalent to the technique used in the separable case: to find the coefficients of the generalized Optimal hyperplane

$$w = \sum_{i=1}^{\ell} \alpha_i y_i x_i,$$

one has to find the parameters $\alpha_i,\ i = 1, ..., \ell$ one that maximize the same quadratic form as in the separable case

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j)$$

under slightly different constraints

$$0 \le \alpha_i \le C, \quad i = 1, \ldots, \ell,$$

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0.$$

As in the separable case, only some of the coefficients $\alpha_i,\ i = 1, ..., \ell$ differ from zero. They determine the support vectors.

Note that if the coefficient C in the functional $\Phi(w,\xi)$ is equal to the optimal value of parameter C^* for minimization of the functional $F_1(\xi)$

$$C = C^*,$$

then the solution to both optimization problems (defined by the functional $F_1(\xi)$ and by the functional $\Phi(w,\xi)$) coincide.

5.6 SUPPORT VECTOR (SV) MACHINES

The Support Vector (SV) machine implements the following idea: it maps the input vectors x into a high-dimensional feature space Z through some nonlinear mapping, chosen *a priori*. In this space, an Optimal separating hyperplane is constructed (Fig. 5.3).

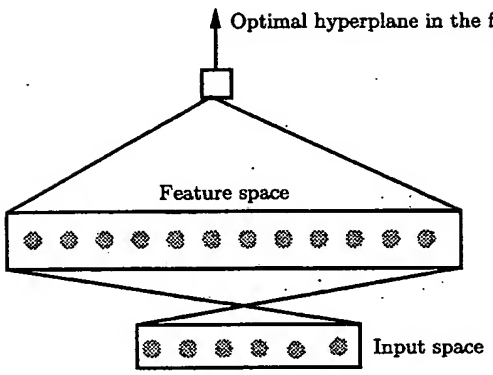


FIGURE 5.3. The SV machine maps the input space into a high-dimensional feature space and then constructs an Optimal hyperplane in the feature space.

Example. To construct a decision surface corresponding to a polynomial of degree two, one can create a feature space Z which has $N = \frac{n(n+3)}{2}$ coordinates of the form

$$z^1 = x^1, \ldots, z^n = x^n, \qquad n \text{ coordinates},$$
$$z^{n+1} = (x^1)^2, \ldots, z^{2n} = (x^n)^2, \qquad n \text{ coordinates},$$
$$z^{2n+1} = x^1 x^2, \ldots, z^N = x^n x^{n-1}, \qquad \frac{n(n-1)}{2} \text{ coordinates},$$

where $x = (x^1, \ldots, x^n)$. The separating hyperplane constructed in this space is a second degree polynomial in the input space.

Two problems arise in the above approach: one conceptual and one technical.

(i) *How to find a separating hyperplane that will generalize well?* (The conceptual problem.)

The dimensionality of the feature space will be huge, and a hyperplane that separates the training data will not necessarily generalize well.[5]

(ii) *How to treat computationally such high-dimensional spaces?* (The technical problem.)

To construct a polynomial of degree 4 or 5 in a 200 dimensional space it is necessary to construct hyperplanes in a billion dimensional feature space. How can this "curse of dimensionality" be overcome?

[5]Recall Fisher's concern about the small amount of data for constructing a quadratic discriminant function in classical discriminant analysis (Section 1.9).

5.6.1 Generalization in High-Dimensional Space

The conceptual part of this problem can be solved by constructing the Optimal hyperplane.

According to Theorem 5.1, if it happens that in the high-dimensional input space one can construct a separating hyperplane with a small value of $[R^2A^2]$, the VC dimension of the corresponding element of the structure will be small, and therefore the generalization ability of the constructed hyperplane will be high.

Furthermore, the following theorem holds.

Theorem 5.2. *If the training vectors are separated by the Optimal hyperplane (or generalized Optimal hyperplane), then the expectation value of the probability of committing an error on a test example is bounded by the ratio of the expectation of the number of support vectors to the number of examples in the training set:*

$$E[P(\text{error})] \leq \frac{E[\text{number of support vectors}]}{(\text{number of training vectors}) - 1}. \tag{5.23}$$

This bound depends neither on the dimensionality of the space, nor on the norm of the vector of coefficients, nor on the bound of the norm of the input vectors. Therefore, if the Optimal hyperplane can be constructed from a small number of support vectors relative to the training set size, the generalization ability will be high — even in an infinite-dimensional space.[6]

5.6.2 Convolution of the Inner Product

However, even if the Optimal hyperplane generalizes well and can theoretically be found, the technical problem of how to treat the high-dimensional feature space remains.

In 1992 it was observed (Boser, Guyon, and Vapnik, 1992) that for constructing the Optimal separating hyperplane in the feature space Z, one

---

[6] One can compare the result of this theorem to result of analysis of the following compression scheme. To construct the Optimal separating hyperplane one only needs to specify among the training data the support vectors and its classification. This requires: $\approx \lceil \lg_2 m \rceil$ bits to specify the number m of support vectors, $\lceil \lg_2 C_\ell^m \rceil$ bits to specify the support vectors; and $\lceil \lg_2 C_m^{m_1} \rceil$ bits to specify representatives of the first class among the support vectors. Therefore for $m << \ell$ and $m_1 \approx m/2$ the compression coefficient is

$$K \sim \frac{m(\lg_2 \ell/m + 1)}{\ell}.$$

The expectation of this coefficient should be compared to the value $Em/(\ell - 1)$ (the right hand side of inequality (5.23)).

does not need to consider the feature space in *explicit form*. One only has to be able to calculate the inner products between support vectors and the vectors of the feature space (Eqs. (5.17) and (5.20)).

Consider a general expression for the inner product in Hilbert space[7]

$$(z_i \cdot z) = K(x, x_i),$$

where z is the image in feature space of the vector x in input space.

According to Hilbert–Schmidt theory, $K(x, x_i)$ can be any symmetric function satisfying the following general conditions (Courant and Hilbert, 1953):

Theorem 5.3. (Mercer) *To guarantee that the symmetric function $K(u, v)$ from L_2 has an expansion*

$$K(u, v) = \sum_{k=1}^{\infty} a_k \psi_k(u) \psi_k(v) \tag{5.24}$$

with positive coefficients $a_k > 0$ (i.e., $K(u, v)$ describes a inner product in some feature space), it is necessary and sufficient that the condition

$$\int\int K(u, v) g(u) g(v)\, du\, dv > 0$$

be valid for all $g \neq 0$ for which

$$\int g^2(u) du < \infty.$$

5.6.3 Constructing SV Machines

The convolution of the inner product allows the construction of decision functions that are nonlinear in the input space

$$f(x) = \text{sign}\left(\sum_{\text{support vectors}} y_i \alpha_i K(x_i, x) - b \right), \tag{5.25}$$

and that are equivalent to linear decision functions in the high-dimensional feature space $\psi_1(x), .., \psi_N(x)$ ($K(x_i, x)$ is a convolution of the inner product for this feature space).

---

[7] This idea was used in 1964 by Aizerman, Braverman, and Rozonoer in their analysis of the convergence properties of the method of Potential functions (Aizerman, Braverman, and Rozonoer, 1964, 1970). It happened at the same time (1965) when the method of the Optimal hyperplane was developed (Vapnik and Chervonenkis 1965). However, combining these two ideas, which lead to the SV machines, was only done in 1992.

To find the coefficients α_i in the separable case (analogously in the non-separable case) it is sufficient to find the maximum of the functional

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j}^{\ell} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{5.26}$$

subject to the constraints

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0,$$

$$\alpha_i \geq 0, \quad i = 1, 2, \ldots, \ell. \tag{5.27}$$

This functional coincides with the functional for finding the Optimal hyperplane, except for the form of the inner products: instead of inner products $(x_i \cdot x_j)$ in Eq. (5.17), we now use the convolution of the inner products $K(x_i, x_j)$.

The learning machines which construct decision functions of the type (5.25) are called *Support Vector (SV) Machines*. (With this name we stress the idea of expanding the solution on support vectors. In SV machines the complexity of the construction depends on the number of support vectors rather than on the dimensionality of the feature space.) The scheme of SV machines is shown in Fig. 5.4.

5.6.4 *Examples of SV Machines*

Using different functions for convolution of the inner products $K(x, x_i)$, one can construct learning machines with different types of nonlinear decision surfaces in input space. Below, we consider three types of learning machines:

(i) Polynomial Learning Machines,

(ii) Radial Basis Functions Machines, and

(iii) Two Layer Neural Networks.

For simplicity we consider here the regime where the training vectors are separated without error.

Note that the support vector machines implement the SRM principle. Indeed, let

$$\Psi(x) = (\psi_1(x), \ldots, \psi_N(x))$$

be a feature space and $w = (w_1, \ldots, w_N)$ be a vector of weights determining a hyperplane in this space. Consider a structure on the set of hyperplanes with elements S_k containing the functions satisfying the conditions

$$R^2 |w|^2 \leq k,$$

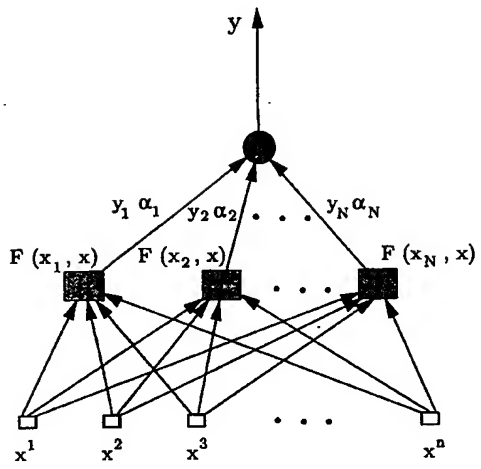


FIGURE 5.4. The two-layer SV machine is a compact realization of an Optimal hyperplane in the high-dimensional feature space Z.

where R is the radius of the smallest sphere that contains the vectors $\Psi(x)$, $|w|$ is the norm of the weights (we use canonical hyperplanes in feature space with respect to the vectors $z = \Psi(x_i)$ where x_i are the elements of the training data).

According to Theorem 5.1 (now applied in the feature space), k gives an estimate of the VC dimension of the set of functions S_k.

The SV machine separates without error the training data

$$y_i\,[(\Psi(x_i)\cdot w)+b]\geq 1,\quad y_i=\{+1,\ -1\},\quad i=1,2,\ldots,\ell,$$

and has a minimal norm $|w|$.

In other words, the SV machine separates the training data using functions from element S_k with the smallest estimate of the VC dimension.

Recall that in the feature space the equality

$$|w_0|^2=\sum_{i}^{\ell}\alpha_i^0\alpha_j^0K(x_i,x_j)y_iy_j \tag{5.28}$$

holds true. To control the generalization ability of the machine (to minimize the probability of test errors) one has to construct the separating

hyperplane that minimizes the functional

$$\Phi(R, w_0, \ell) = \frac{R^2|w_0|^2}{\ell}. \tag{5.29}$$

Indeed, for separating hyperplanes the probability of test errors with probability $1 - \eta$ is bounded by the expression

$$\mathcal{E} = 4\frac{h(\ln\frac{2\ell}{h} + 1) - \ln\eta/4}{\ell}.$$

The right-hand side of $\mathcal{E}$ attains its minimum when h/ℓ is minimal. We estimate the minimum of h/ℓ by estimating h by $h_{est} = R^2|w_0|^2$. To estimate this functional it is sufficient to estimate $|w_0|^2$ (say by expression (5.28)) and estimate R^2 by finding

$$R^2 = R^2(K) = \min_a \max_{x_i} [K(x_i, x_i) + K(a, a) - 2K(x_i, a)]. \tag{5.30}$$

Polynomial Learning Machine

To construct polynomial decision rules of degree d, one can use the following function for convolution of the inner product:

$$K(x, x_i) = [(x \cdot x_i) + 1]^d. \tag{5.31}$$

This symmetric function satisfies the conditions of Theorem 5.3, therefore it describes a convolution of the inner product in the feature space that contains all products $x_i \cdot x_j \cdot x_k$ up to degree d. Using the described technique, one constructs a decision function of the form

$$f(x, \alpha) = \text{sign}\left(\sum_{\text{support vectors}} y_i\alpha_i[(x_i \cdot x) + 1]^d - b\right),$$

which is a factorization of d-dimension polynomials in n-dimensional input space.

In spite of the very high dimensionality of the feature space (polynomials of degree d in n-dimensional input space have $O(n^d)$ free parameters) the estimate of the VC dimension of the subset of polynomials that solve real life problems can be low.

As described above to estimate the VC dimension of the element of the structure from which the decision function is chosen, one has only to estimate the radius R of the smallest sphere that contains the training data, and the norm of weights in feature space (Theorem 5.1).

Note that both the radius $R = R(d)$ and the norm of weights in the feature space depends on the degree of the polynomial.

This gives the opportunity to choose the best degree of polynomial for the given data.

To make a *local polynomial* approximation in the neighborhood of a point of interest x_0, let us consider the hard threshold neighborhood function (4.16). According to the theory of local algorithms, one chooses a ball with radius R_β around point x_0 in which ℓ_β elements of the training set fall, and then using only these training data, constructs the decision function that minimizes the probability of errors in the chosen neighborhood. The solution to this problem is a radius R_β that minimizes the functional

$$\Phi(R_\beta, w_0, \ell_\beta) = \frac{R_\beta^2 |w_0|^2}{\ell_\beta} \tag{5.32}$$

(the parameter $|w_0|$ depends on the chosen radius as well). This functional describes a trade-off between the chosen radius R_β, the value of the minimum of the norm $|w_0|$, and the number of training vectors ℓ_β that fall into radius R_β.

Radial Basis Function Machines

Classical Radial Basis Function (RBF) Machines use the following set of decision rules:

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} a_i K_\gamma(|x - x_i|) - b\right), \tag{5.33}$$

where $K_\gamma(|x - x_i|)$ depends on the distance $|x - x_i|$ between two vectors. For the theory of RBF machines see (Micchelli, 1986), (Powell, 1992).

The function $K_\gamma(|x - x_i|)$ is for any fixed γ, a non-negative monotonic function; it tends to zero as z goes to infinity. The most popular function of this type is

$$K_\gamma(|x - x_i|) = \exp\{-\gamma |x - x_i|^2\}. \tag{5.34}$$

To construct the decision rule (5.33) one has to estimate

(i) The value of the parameter γ,

(ii) the number N of the centers x_i,

(iii) the vectors x_i, describing the centers,

(iv) the value of the parameters a_i.

In the classical RBF method the first three steps (determining the parameters γ, N, and vectors (centers) x_i, $i = 1, \ldots, N$) are based on heuristics and only the fourth step (after finding these parameters) is determined by minimizing the empirical risk functional.

The radial function can be chosen as a function for the convolution of the inner product for a SV machine. In this case, the SV machine will construct a function from the set (5.33). One can show (Aizerman, Braverman, and

Rozonoer, 1964, 1970) that radial functions (5.34) satisfy the condition of Theorem 5.3.

In contrast to classical RBF methods, in the SV technique all four types of parameters are chosen to minimize the bound on probability of test error by controlling the parameters R, w_0 in the functional (5.29). By minimizing the functional (5.29) one determines

(i) N, the number of support vectors,

(ii) x_i, (the pre-images of) support vectors;

(iii) $a_i = \alpha_i y_i$, the coefficients of expansion, and

(iv) γ, the width parameter of the kernel-function.

Two-Layer Neural Networks

Finally, one can define two-layer neural networks by choosing kernels:

$$K(x, x_i) = S[v(x \cdot x_i) + c],$$

where $S(u)$ is a sigmoid function. In contrast to kernels for polynomial machines or for radial basis function machines that alway satisfy Mercer conditions, the sigmoid kernel $\tanh(vu+c)$, $|u| \leq 1$, satisfies Mercer conditions only for some values of parameters v, c. For these values of parameters one can construct SV machines implementing the rules:

$$f(x, \alpha) = \text{sign}\left\{\sum_{i=1}^{N} \alpha_i S(v(x \cdot x_i) + c) + b\right\}.$$

Using the technique described above, the following are found automatically:

(i) The architecture of the two layer machine, determining the number N of hidden units (the number of support vectors),

(ii) the vectors of the weights $w_i = x_i$ in the neurons of the first (hidden) layer (the support vectors), and

(iii) the vector of weights for the second layer (values of α).

5.7 EXPERIMENTS WITH SV MACHINES

In the following we will present two types of experiments constructing the decision rules in the pattern recognition problem[8]:

[8]The experiments were conducted in the Adaptive System Research Department, AT&T Bell Laboratories.

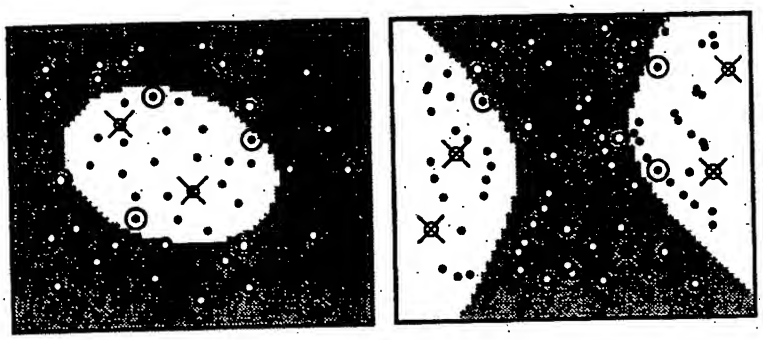


FIGURE 5.5. Two classes of vectors are represented in the picture by black and white balls. The decision boundaries were constructed using an inner product of polynomial type with $d = 2$. In the pictures the examples cannot be separated without errors; the errors are indicated by crosses and the support vectors by double circles.

(i) Experiments in the plane with artificial data that can be visualized, and

(ii) experiments with real-life data.

5.7.1 *Example in the Plane*

To demonstrate the SV technique we first give an artificial example (Fig. 5.5).

The two classes of vectors are represented in the picture by black and white balls. The decision boundaries were constructed using a inner product of polynomial type with $d = 2$. In the pictures the examples cannot be separated without errors; the errors are indicated by crosses and the support vectors by double circles.

Notice that in both examples the number of support vectors is small relative to the number of training data and that the number of training errors is minimal for polynomials of degree two.

5.7.2 *Handwritten Digit Recognition*

Since the first experiments of Rosenblatt, the interest in the problem of learning to recognize handwritten digits has remained strong. In the following we describe results of experiments on learning the recognition of handwritten digits using different SV machines. We also compare these results to results obtained by other classifiers. In these experiments, the U.S. Postal Service database (LeCun et *al.*, 1990) was used. It contains 7,300 training patterns and 2,000 test patterns collected from real-life zip–codes. The resolution of the database is 16×16 pixels, therefore the dimensionality

Classifier	Raw error%
Human performance	2.5
Decision tree, C4.5	16.2
Best two-layer neural network	5.9
Five-layer network (LeNet 1)	5.1

TABLE 5.1. Human performance and performance of the various learning machines, solving the problem of digit recognition on U.S. Postal Service data.

of the input space is 256. Figure 5.6 gives examples from this data-base.

Table 5.1 describes the performance of various classifiers, solving this problem.[9]

For constructing the decision rules three types of SV machines were used[10]:

(i) A polynomial machine with convolution function:

$$K(x, x_i) = \left(\frac{(x \cdot x_i)}{256}\right)^d, \quad d = 1, .., 7.$$

(ii) A radial basis function machine with convolution function:

$$K(x, x_i) = \exp\left\{-\frac{(x - x_i)^2}{256\sigma^2}\right\}.$$

(iii) A two layers neural network machine with convolution function:

$$K(x, x_i) = \tanh\left(\frac{b(x \cdot x_i)}{256} - c\right).$$

All machines constructed ten classifiers, each one separating one class from the rest. The ten class classification was done by choosing the class with the largest classifier output value.

The results of these experiments are given in Table 5.2. For different types of SV machines, Table 5.2 shows: the best parameters for the machines (column 2), the average (over one classifier) of the number of support vectors, and the performance of machine.

[9]The result of human performance was reported by J. Bromley and E. Säckinger; the result of C4.5 was obtained by C. Cortes; the result for the two layer neural net was obtained by B. Schölkopf; the results for the special purpose neural network architecture with five layers (LeNet 1), was obtained by Y. LeCun *et al.*

[10]The results were obtained by C. Burges, C. Cortes, and B. Schölkopf.

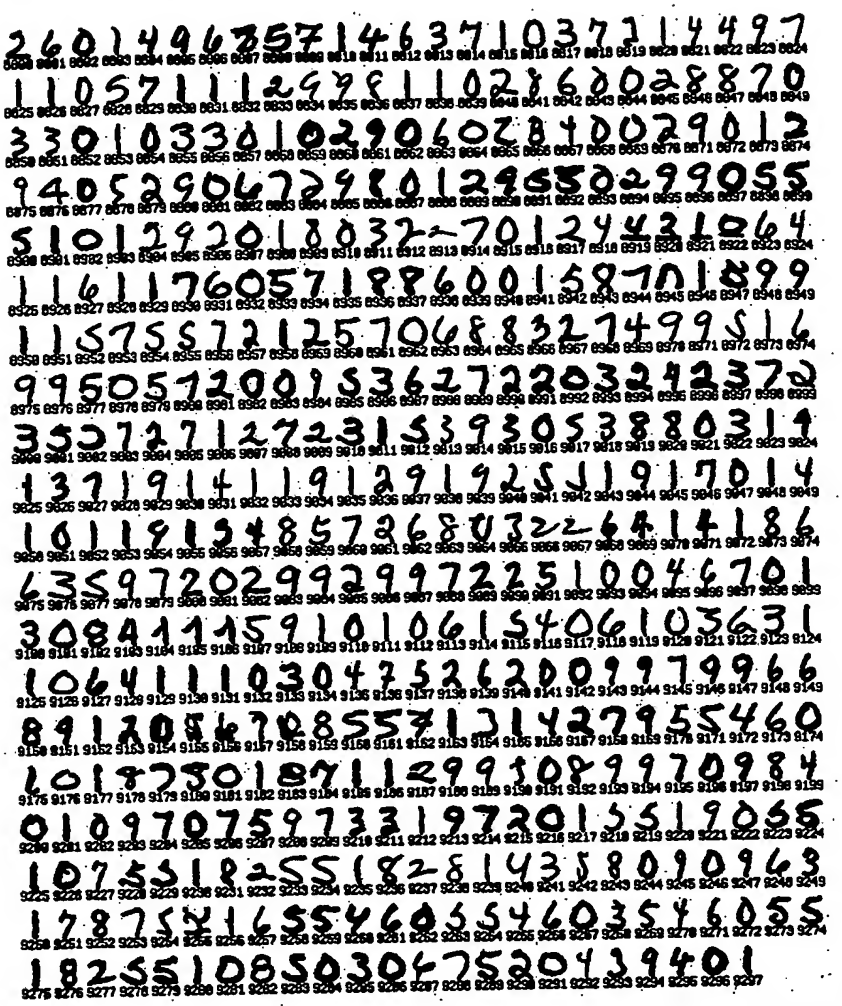


FIGURE 5.6. Examples of patterns (with labels) from the U.S. Postal Service database.

Type of SV classifier	Parameters of classifier	Number of support vectors	Raw error
Polynomials	d=3	274	4.0
RBF classifiers	$\sigma^2 = 0.3$	291	4.1
Neural network	$b = 2,\ c = 1$	254	4.2

TABLE 5.2. Results of digit recognition experiments with various SV machines using the U.S. Postal Service database. The number of support vectors means the average per classifier.

	Poly	RBF	NN	Common
total# of sup.vect.	1677	1727	1611	1377
% of common sup. vect.	82	80	85	100

TABLE 5.3. Total number (in ten classifiers) of support vectors for various SV machines and percentage of common support vectors.

Note that for this problem, all types of SV machines demonstrate approximately the same performance. This performance is better than the performance of any other type of learning machine solving the digit recognition problem by constructing the entire decision rules on the basis of the U.S. Postal Service database.[11]

In these experiments one important singularity was observed: different types of SV machines use approximately the same set of support vectors. The percentage of common support vectors for three different classifiers exceeded 80%.

Table 5.3 describes the total number of different support vectors for ten classifiers of different machines: Polynomial machine (Poly), Radial Basis Function machine (RBF), and Neural Network machine (NN). It shows also the number of common support vectors for all machines.

[11]Note that using a local approximation approach described in Section 5.7 (that does not construct entire decision rule but approximates the decision rule in any point of interest) one can obtain a better result: 3.3% error rate (L. Bottou and V Vapnik, 1992).

The best results for this database, 2.7% was obtained by P. Simard, Y. LeCun, and J. Denker without using any learning methods. They suggested a special method of elastic matching with 7200 templates using a smart concept of distance (so-called Tangent distance) that takes into account invariance with respect to small translations, rotations, distortions, and so on (P. Simard, Y. LeCun, and J. Denker, 1993).

	Poly	RBF	NN
Poly	100	84	94
RBF	87	100	88
NN	91	82	100

TABLE 5.4. Percentage of common (total) support vectors for two SV machines.

Table 5.4 describes the percentage of support vectors of the classifier given in the columns contained in the support vectors of the classifier given in the rows.

This fact, if it holds true for a wide class of real-life problems, is very important.

5.7.3 Some Important Details

In this subsection we give some important details on solving the digit recognition problem using a polynomial SV machine.

The training data are not linearly separable. The total number of misclassifications on the training set for linear rules is equal to 340 ($\approx$ 5% errors). For second degree polynomial classifiers the total number of misclassifications on the training set is down to four. These four mis-classified examples (with desired labels) are shown in Fig. 5.7.Starting with polynomials of degree three, the training data are separable.

Table 5.5 describes the results of experiments using decision polynomials (ten polynomials, one per classifier in one experiment) of various degrees. The number of support vectors shown in the table is a mean value per classifier.

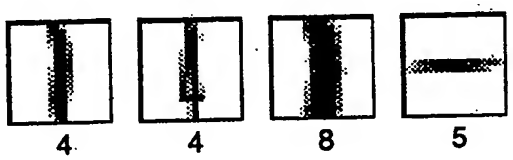


FIGURE 5.7. Labeled examples of training errors for the second degree polynomials.

degree of polynomial	dimensionality of feature space	support vectors	raw error
1	256	282	8.9
2	~ 33000	227	4.7
3	$\sim 1 \times 10^6$	274	4.0
4	$\sim 1 \times 10^9$	321	4.2
5	$\sim 1 \times 10^{12}$	374	4.3
6	$\sim 1 \times 10^{14}$	377	4.5
7	$\sim 1 \times 10^{16}$	422	4.5

TABLE 5.5. Results of experiments with polynomials of the different degrees.

Note that the number of support vectors increases slowly with the degree of the polynomials. The seventh degree polynomial has only 50% more support vectors than the third degree polynomial.[12]

The dimensionality of the feature space for a seventh degree polynomial is however 10^{10} times larger than the dimensionality of the feature space for a third degree polynomial classifier. Note that the performance does not change significantly with increasing dimensionality of the space — indicating no overfitting problems.

To choose the degree of the best polynomials for one specific classifier we estimate the VC dimension (using the estimate $[R^2A^2]$) for all constructed polynomials (from degree two up to degree seven) and choose the one with the smallest estimate of the VC dimension. In this way we found the ten best classifiers (with different degrees of polynomials) for the ten two-class problems. These estimates are shown on Fig. 5.8 where for all ten two-class decision rules, the estimated VC dimension, is plotted versus the degree of the polynomials.The question is:

Do the polynomials with the smallest estimate of the VC dimension provide the best classifier?

To answer this question we constructed Table 5.6 which describes the performance of the classifiers for each degree of polynomial.

Each row describes one two–class classifier separating one *digit* (stated in the first column) from the all ather digits.

The remaining columns contain:

deg.: the degree of the polynomial as chosen (from two up to seven) by the described procedure,

[12]The relatively high number of support vectors for the linear separator is due to nonseparability: the number 282 includes both support vectors and misclassified data.

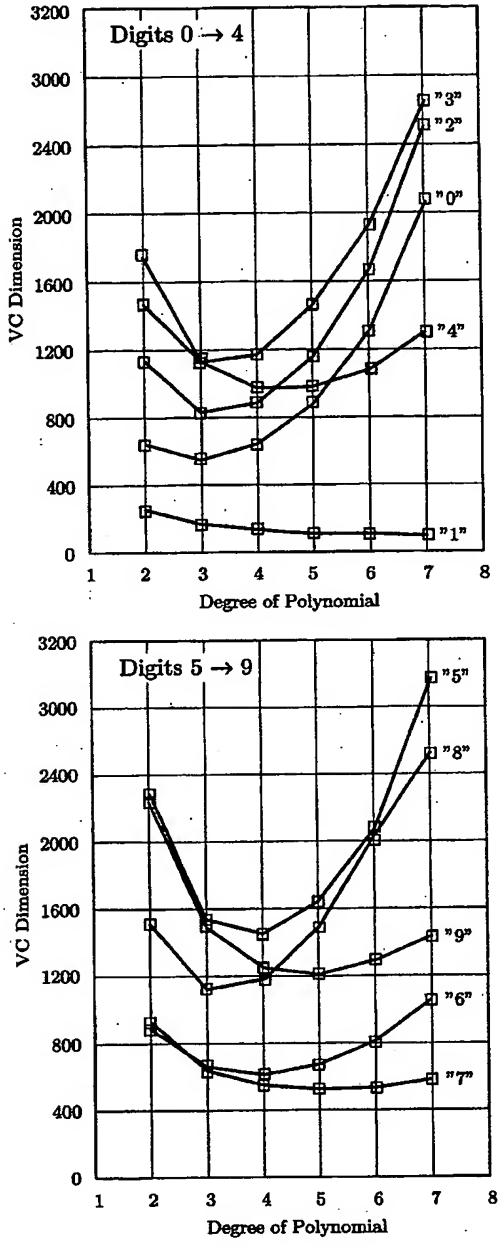


FIGURE 5.8. The estimate of the VC dimension of the best element of the structure (defined on the set of canonical hyperplanes in the corresponding feature space) versus the degree of polynomial for various two-class digit recognition problems (denoted digit versus the rest).

	Chosen classifier			Number of test errors						
Digit	deg.	dim.	$h_{est.}$	1	2	3	4	5	6	7
0	3	$\sim 10^6$	530	36	14	**11**	11	11	12	17
1	7	$\sim 10^{16}$	101	17	15	14	11	10	10	**10**
2	3	$\sim 10^6$	842	53	32	**28**	26	28	27	32
3	3	$\sim 10^6$	1157	57	25	**22**	22	22	22	23
4	4	$\sim 10^9$	962	50	32	32	**30**	30	29	33
5	3	$\sim 10^6$	1090	37	20	**22**	24	24	26	28
6	4	$\sim 10^9$	626	23	12	12	**15**	17	17	19
7	5	$\sim 10^{12}$	530	25	15	12	10	**11**	13	14
8	4	$\sim 10^9$	1445	71	33	28	**24**	28	32	34
9	5	$\sim 10^{12}$	1226	51	18	15	11	**11**	12	15

TABLE 5.6. Experiments on choosing the best degree of polynomial.

dim.: the dimensionality of the corresponding feature space, which is also the maximum possible VC dimension for linear classifiers in that space,

$h_{est.}$: the VC dimension estimate for the chosen polynomial, (which is much smaller than the number of free parameters),

Number of test errors: the number of test errors, using the constructed polynomial of corresponding degree; the boxes show the number of errors for the chosen polynomial.

Thus, Table 5.5 shows that for the SV polynomial machine there are no overfitting problems with increasing degree of polynomials, while Table 5.6 shows that even in situations where the difference between the best and the worst solutions is small (for polynomials starting from degree two up to degree seven), the theory gives a method for approximating the best solutions (finding the best degree of the polynomial).

Note also that Table 5.6 demonstrates that the problem is essentially nonlinear. The difference in the number of errors between the best polynomial classifier and the linear classifier can be as much as a factor of four (for digit 9).

5.8 REMARKS ON SV MACHINES

The quality of any learning machine is characterized by three main components:

(i) *How universal is the learning machine?*
How rich is the set of functions that it can approximate?

(ii) *How well can the machine generalize?*
How close is the upper bound on the error rate that this machine achieves (implementing a given set of functions and a given structure on this set of functions) to the smallest possible?

(iii) *How fast does the learning process for this machine converge?*
How many operations does it take to find the decision rule, using a given number of observations?

We address these in turn below.

(i) SV machines implement the sets of functions

$$f(x, \alpha, w) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i K(x, w_i) - b\right), \tag{5.35}$$

where N is any integer ($N < \ell$), α_i, $i = 1, ..., N$ are any scalars and w_i, $i = 1, .., N$ are any vectors. The kernel $K(x, w)$ can be any symmetric function satisfying the conditions of Theorem 5.3.

As was demonstrated, the best guaranteed risk for these sets of functions is achieved when the vectors of weights $w_1, ...w_N$ are equal to some of the vectors x from the training data (support vectors).

Using the set of functions

$$\bar{f}(x, \alpha, w) = \sum_{\text{support vectors}} y_i \alpha_i K(x, w_i) - b$$

with convolutions of polynomial, radial basis function, or neural network type, one can approximate a continuous function to any degree of accuracy.

Note that for the SV machine one does not need to construct the architecture of the machine by choosing *a priori* the number N (as is necessary in classical neural networks or in classical radial basis function machines).

Furthermore, by changing only the function $K(x, w)$ in the SV machine one can change the type of learning machine (the type of approximating functions).

(ii) SV machines minimize the upper bound on the error rate for the structure given on a set of functions in a feature space. For the best solution it is *necessary* that the vectors w_i in Eq. (5.35) coincide with some vectors of the training data (support vectors[13]). SV machines find the functions

[13]This assertion is a direct corollary of the necessity of the Kühn–Tucker conditions for solving the quadratic optimization problem described in Section 5.4. The Kühn–Tucker conditions are necessary and sufficient for the solution of this problem.

from the set (5.35) that separate the training data and belong to the subset with the smallest bound of the VC dimension. (In the more general case they minimize the bound of the risk (5.1).)

(iii). Finally, to find the desired function, the SV machine has to maximize a non-positive quadratic form in the non-negative quadrant. This problem is a particular case of a special quadratic programming problem: to maximize a non-positive quadratic form $Q(x)$ with bounded constraints

$$a_i \le x^i \le b_i, \quad i = 1, \ldots, n,$$

where x^i, $i = 1, \ldots, n$ are the coordinates of the vector x and a_i, b_i are given constants. For this specific quadratic programming problem fast algorithms exist.

5.9 SV MACHINES FOR THE REGRESSION ESTIMATION PROBLEM

5.9.1 *ε-Insensitive Loss-Function*

In Section 1.3.2 to describe the problem of approximation of the supervisor's rule $F(y|x)$ for the case where y is real valued we considered a quadratic loss-function

$$L(y, f(x, \alpha)) = (y - f(x, \alpha))^2. \tag{5.36}$$

Using the ERM inductive principle and this loss-function one obtains a function that gives the best least squares approximation to the data. Under conditions where y is the result of measuring a function with normal additive noise (see Section 1.7.3) (and for the ERM principle) this loss-function provides also the best approximation to the regression.

It is known, however, that if additive noise is generated by another law, the optimal approximation to the regression (for the ERM principle) gives another loss-function (associated with this law).

In 1964 Huber developed a theory that allows us to define the best loss-function for the problem of regression estimation on the basis of the ERM principle if one has only general information about the model of the noise. In particular, he showed that if one only knows that the density $p(x)$ describing the noise is a symmetric convex function possessing second derivatives, then the best minimax approximation to regression (the best approximation for

the worst possible $p(x)$) provides the loss-function[14]

$$L(y, f(x,\alpha)) = |y - f(x,\alpha)|. \tag{5.37}$$

Minimizing the empirical risk with respect to this loss-function is called the *least modulus* method. It defines the so-called *robust regression* function.

We consider a slightly more general type of loss-function than (5.37), the so-called linear loss-function with insensitive zone:

$$|y - f(x,\alpha)|_\varepsilon = \begin{cases} \varepsilon, & \text{if } |y - f(x,\alpha)| \le \varepsilon \\ |y - f(x,\alpha)|, & \text{otherwise.} \end{cases}, \tag{5.38}$$

This loss-function describes the ε-insensitive model: the loss is equal to ε if the discrepancy between the predicted and actual value is less than ε and is equal to the discrepancy otherwise. The loss-function (5.37) is a particular case of this loss-function for $\varepsilon = 0$.

5.9.2 Minimizing the Risk Using Convex Optimization Procedure

The support vector type approximation to regression takes place if:

(i) One estimates the regression in the set of linear functions

$$f(x,\alpha) = (w \cdot x) + b.$$

(ii) One defines the problem of regression estimation as the problem of risk minimization with respect to an ε-insensitive ($\varepsilon \ge 0$) loss-function (5.38).

(iii) One minimizes the risk using the *SRM principle*, where elements of the structure S_n are defined by inequality

$$(w \cdot w) \le c_n. \tag{5.39}$$

1. Indeed, suppose we are given training data

$$(x_1, y_1), \ldots, (x_\ell, y_\ell).$$

Then the problem of finding the w_ℓ and b_ℓ that minimize the empirical risk

$$R_{\text{emp}}(w,b) = \frac{1}{\ell}\sum_{i=1}^{\ell} |y_i - (w \cdot x_i) - b|_\varepsilon$$

[14]This is an extreme case where one has minimal information about an unknown density. Huber described also the intermediate cases where the unknown density is a mixture of some given density and any density from a described set of densities, taken in proportion ϵ and $1 - \epsilon$ (Huber, 1964).

under constraint (5.39) is equivalent to the problem of finding the pair w, b that minimizes the quantity defined by slack variables ξ_i, ξ_i^*, $i = 1, ..., \ell$

$$F(\xi, \xi^*) = \left(\sum_{i=1}^{\ell} \xi_i^* + \sum_{i=1}^{\ell} \xi_i \right) \tag{5.40}$$

under constraints

$$\begin{array}{rcll} y_i - (w \cdot x_i) - b & \leq & \varepsilon + \xi_i^*, & i = 1, ..., \ell, \\ (w \cdot x_i) + b - y_i & \leq & \varepsilon + \xi_i, & i = 1, ..., \ell, \\ \xi_i^* & \geq & 0, & i = 1, ..., \ell, \\ \xi_i & \geq & 0, & i = 1, ..., \ell, \end{array} \tag{5.41}$$

and constraint (5.39).

As before to solve the optimization problem with constraints of inequality type one has to find a saddle point of the Lagrange functional

$$L(w, \xi^*, \xi; \alpha^*, \alpha, C^*, \gamma, \gamma^*) = \sum_{i=1}^{\ell} (\xi_i^* + \xi_i) - \sum_{i=1}^{\ell} \alpha_i \left[y_i - (w \cdot x_i) - b + \varepsilon + \xi_i \right] -$$

$$\sum_{i=1}^{\ell} \alpha_i^* \left[(w \cdot x_i) + b - y_i + \varepsilon + \xi_i^* \right] - \frac{C^*}{2} (c_n - (w \cdot w)) - \sum_{i=1}^{\ell} (\gamma_i^* \xi_i^* + \gamma_i \xi_i). \tag{5.42}$$

(Minimum with respect to elements w, b, ξ_i^*, and ξ_i^* and maximum with respect to Lagrange multipliers $C^* \geq 0$, $\alpha_i^* \geq 0$, $\alpha_i \geq 0$, $\gamma_i^* \geq 0$, and $\gamma_i \geq 0$, $i = 1, ..., \ell$.)

Minimization with respect to w, b and ξ_i^*, ξ_i implies the following three conditions:

$$w = \sum_{i=1}^{\ell} \frac{\alpha_i^* - \alpha_i}{C^*} x_i, \tag{5.43}$$

$$\sum_{i=1}^{\ell} \alpha_i^* = \sum_{i=1}^{\ell} \alpha_i, \tag{5.44}$$

$$0 \leq \alpha_i^* \leq 1, \quad i = 1, ..., \ell,$$

$$0 \leq \alpha_i \leq 1, \quad i = 1, ..., \ell, \tag{5.45}$$

Putting (5.43) and (5.44) into (5.42) one obtains that, for solution of this optimization problem, one has to find the maximum of the convex functional

$$W(\alpha, \alpha^*, C^*) = -\varepsilon \sum_{i=1}^{\ell} (\alpha_i^* + \alpha_i) + \sum_{i=1}^{\ell} y_i (\alpha_i^* - \alpha_i)$$

$$-\frac{1}{2C^*}\sum_{i,j=1}^{\ell}(\alpha_i^*-\alpha_i)(\alpha_j^*-\alpha_j)(x_i\cdot x_j)-\frac{c_nC^*}{2} \quad (5.46)$$

subject to constraints (5.44), (5.45) and constraint

$$C^* \geq 0.$$

As in pattern recognition here only some of the parameters in expansion (5.43)

$$\beta_i = \frac{\alpha_i^*-\alpha_i}{C^*}, \quad i=1,...,\ell$$

differ from zero. They define the support vectors of the problem.

2. One can reduce the convex optimization problem of finding the vector w to a quadratic optimization problem, if instead of minimizing the functional (5.40), subject to constraints (5.41) and (5.39), one minimizes

$$\Phi(w,\xi^*,\xi)=\frac{1}{2}(w\cdot w)+C\left(\sum_{i=1}^{\ell}\xi_i^*+\sum_{i=1}^{\ell}\xi_i\right)$$

(with given value C) subject to constraints (5.41). In this case to find the desired vector

$$w=\sum_{i=1}^{\ell}(\alpha_i^*-\alpha_i)x_i,$$

one has to find coefficients α_i^*, α_i, $i=1,...,\ell$ that maximize the quadratic form

$$W(\alpha,\alpha^*)=-\varepsilon\sum_{i=1}^{\ell}(\alpha_i^*+\alpha_i)+\sum_{i=1}^{\ell}y_i(\alpha_i^*-\alpha_i)-\frac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i^*-\alpha_i)(\alpha_j^*-\alpha_j)(x_i\cdot x_j) \quad (5.47)$$

subject to constraints

$$\sum_{i=1}^{\ell}\alpha_i^*=\sum_{i=1}^{\ell}\alpha_i,$$

$$0\leq\alpha_i^*\leq C, \quad i=1,...,\ell,$$

$$0\leq\alpha_i\leq C, \quad i=1,...,\ell.$$

As in the pattern recognition case, the solution to these two optimization problems coincide if $C=C^*$.

One can show that for any $i=1,...,\ell$ the equality

$$\alpha_i^*\times\alpha_i=0$$

holds true. Therefore, for the particular case where $\varepsilon = 0$ and $y_i \in \{-1, 1\}$ the considered optimization problems coincide with those described for pattern recognition in Section 5.5.1.

To derive the bound on the generalization of the SV machine, suppose that distribution $F(x, y) = F(y|x)F(x)$ is such that for any fixed w, b the corresponding distribution of the random variable $|y - (w \cdot x) - b|_\varepsilon$ has a "light tail" (see Section 3.4):

$$\sup_{w,d} \frac{\sqrt[p]{E|y - (w \cdot x) - b|_\varepsilon^p}}{E|y - (w \cdot x) - b|_\varepsilon} \le \tau, \quad p > 2.$$

Then according to equation (3.30) one can assert that the solution w_ℓ, b_ℓ of the optimization problem, provides a risk (with respect to loss function (5.38)) such that with probability at least $1 - \eta$ the bound

$$R(w_\ell, b_\ell) \le \varepsilon + \frac{R_{emp}(w_\ell, b_\ell) - \varepsilon}{\left(1 - a(p)\tau\sqrt{\mathcal{E}}\right)_+},$$

holds true, where

$$a(p) = \sqrt[p]{\frac{1}{2}\left(\frac{p-1}{p-2}\right)^{p-1}}$$

and

$$\mathcal{E} = 4\frac{h_n\left(\ln\frac{2\ell}{h_n} + 1\right) - \ln(\eta/4)}{\ell}.$$

Here h_n is the VC dimension of the set of functions

$$S_n = \{|y - (w \cdot x) - b|_\varepsilon : \ (w \cdot w) \le c_n\}.$$

5.9.3 SV Machine with Convolved Inner Product

Constructing the best approximation of the form

$$f(x; v, \beta) = \sum_{i=1}^{N} \beta_i K(x, v_i) + b \tag{5.48}$$

where β_i, $i = 1, .., N$ are scalars, v_i, , $i = 1, ..., N$ are vectors, and $K(\cdot, \cdot)$ is a given function satisfying Mercer's conditions, is analogous to constructing a linear approximation. It can be conducted both by solving a convex optimization problem and by solving a quadratic optimization problem.

1. Using the convex optimization approach one evaluates coefficients β_i, $i = 1, ..., \ell$ in (5.48) as

$$\beta_i = \frac{\alpha_i^* - \alpha_i}{C^*}, \quad i = 1, ..., \ell,$$

where α_i^*, α_i, C are parameters that maximize the function

$$W = -\varepsilon \sum_{i=1}^{\ell}(\alpha_i^* + \alpha_i) + \sum_{i=1}^{\ell} y_i(\alpha_i^* - \alpha_i) - \frac{1}{2C^*} \sum_{i,j=1}^{\ell} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)K(x_i \cdot x_j) - \frac{c_n C^*}{2}$$

subject to the constraint

$$\sum_{i=1}^{\ell} \alpha_i^* = \sum_{i=1}^{\ell} \alpha_i$$

and to constraints

$$0 \le \alpha_i^* \le 1, \quad 0 \le \alpha_i \le 1, \quad i = 1, ..., \ell,$$
$$0 \le \alpha_i \le 1, \quad i = 1, ..., \ell,$$

and

$$C^* \ge 0.$$

2. Using the quadratic optimization approach one evaluates vector w (5.48) with coordinates

$$\beta_i = \alpha_i^* - \alpha_i, \quad i = 1, ..., \ell,$$

where α_i^*, α_i are parameters that maximize the function

$$W = -\varepsilon \sum_{i=1}^{\ell}(\alpha_i^* + \alpha_i) + \sum_{i=1}^{\ell} y_i(\alpha_i^* - \alpha_i) - \frac{1}{2} \sum_{i,j=1}^{\ell} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)K(x_i \cdot x_j)$$

subject to the constraint

$$\sum_{i=1}^{\ell} \alpha_i^* = \sum_{i=1}^{\ell} \alpha_i$$

and to constraints

$$0 \le \alpha_i^* \le C, \quad i = 1, ..., \ell,$$
$$0 \le \alpha_i \le C, \quad i = 1, ..., \ell.$$

Choosing different kernels $K(\cdot, \cdot)$ satisfying Mercer's condition one constructs different types of learning machine. In particular, the kernel

$$K(x, x_i) = [(x \cdot x_i) + 1]^r$$

gives a polynomial machine.

By controlling two parameters c_n and ε (C and ε in the quadratic optimization approach) one can control the generalization ability, even for high degree polynomials in a high-dimensional space.

Informal Reasoning and Comments 5

5.10 THE ART OF ENGINEERING VERSUS FORMAL INFERENCE

The existence of neural networks can be considered a challenge for theoreticians.

From the formal point of view one cannot guarantee that neural networks generalize well, since according to theory, in order to control generalization ability one should control two factors: the value of the empirical risk and the value of the confidence interval. Neural networks, however, cannot control either of the two.

Indeed, to minimize the empirical risk, a neural network must minimize a functional that has many local minima. Theory offers no constructive way to prevent ending up with unacceptable local minima. In order to control the confidence interval one has first to construct a structure on the set of functions that the neural network implements and then to control capacity using this structure. There are no accurate methods to do this for neural networks.

Therefore from the formal point of view it seems that there should be no question as to what type of machine should be used for solving real-life problems.

The reality however is not so straightforward. The designers of neural networks compensate the mathematical shortcomings with the high art of engineering. Namely, they incorporate various heuristic algorithms that

make it possible to attain reasonably local minima using a reasonably small number of calculations.

Moreover, for given problems they create special network architectures which both have an appropriate capacity and contain "useful" functions for solving the problem. Using these heuristics, neural networks demonstrate surprisingly good results.

In Chapter 5, describing the best results for solving the digit recognition problem using the U.S. Postal Service database by constructing an entire (not local) decision rule we gave two figures:

> 5.1% error rate for the neural network LeNet 1 (designed by Y. LeCun),
>
> 4.0% error rate for a polynomial SV machine.

We also mentioned the two best results:

> 3.3% error rate for the local learning approach, and the record
>
> 2.7% error rate for tangent distance matching to templates given by the training set.

In 1993, responding to the community's need for benchmarking, the U.S. National Institute of Standard and Technology (NIST) provided a database of handwritten characters containing 60,000 training images and 10,000 test data, where characters are described as vectors in $20 \times 20 = 400$ pixel space.

For this database a special neural network (LeNet 4) was designed. The following is how the article reporting the benchmark studies (Léon Bottou *et al*, 1994) describes the construction of LeNet 4:

> "For quite a long time, LeNet 1 was considered the state of the art. The local learning classifier, the SV classifier, and tangent distance classifier were developed to improve upon LeNet 1 — and they succeeded in that. However, they in turn motivated a search for an improved neural network architecture. This search was guided in part by estimates of the capacity of various learning machines, derived from measurements of the training and test error (on the large NIST database) as a function of the number of training examples.[15] We discovered that more capacity was needed. Through a series of experiments in architecture, combined with an analysis of the characteristics of recognition errors, LeNet 4 was crafted."

In these benchmarks, two learning machines that construct entire decision rules:

---

[15] V. Vapnik, E. Levin, and Y. LeCun (1994) "Measuring the VC dimension of a learning machine," *Neural Computation*, 6(5), pp. 851-876.

(i) LeNet 4,

(ii) Polynomial SV machine (polynomial of degree four)

provided the same performance: 1.1% test error[16].

The local learning approach and tangent distance matching to 60,000 templates also gave the same performance: 1.1% test error.

Recall that for a small (U.S. Postal Service) database the best result (by far) was obtained by the tangent distance matching method which uses *a priori* information about the problem (incorporated in the concept of tangent distance). As the number of examples increases to 60,000 the advantage of *a priori* knowledge decreased. The advantage of the local learning approach also decreased with the increasing number of observations.

LeNet 4, crafted for the NIST database demonstrated remarkable improvement in performance comparing to LeNet 1 (which has 1.7% test errors for the NIST database[17]).

The standard polynomial SV machine also did a good job. We continue the quotation (Léon Bottou, *et al*, 1994):

> "The SV machine has excellent accuracy, which is most remarkable, because unlike the other high performance classifiers it *does not include knowledge about the geometry of the problem.* In fact this classifier would do just as well if the image pixel were encrypted, e.g., by a fixed random permutation."

However, the performance achieved by these learning machines is not the record for the NIST database. Using models of characters (the same that was used for constructing the tangent distance) and 60,000 examples of training data, H. Drucker, R. Schapire, and P. Simard generated more than 1,000,000 examples which they used to train three LeNet 4 neural networks, combined in the special "boosting scheme" (Drucker, Schapire, and Simard, 1993) which achieved a 0.7% error rate.

Now the SV machines have a challenge — to cover this gap (between 1.1% to 0.7%). Probably the use of only brute force SV machines and 60,000 training examples will not be sufficient to cover the gap. Probably one has to incorporate some *a priori* information about the problem at hand.

There are several ways to do this. The simplest one is use the same 1,000,000 examples (constructed from the 60,000 NISTs prototypes). However, it is more interesting to find a way for directly incorporating the

---

[16]Unfortunately one cannot compare these results to the results described in Chapter 5. The digits from the NIST database are "easier" for recognition than the ones from U.S. Postal Service database.

[17]Note that LeNet 4 has an advantage for large 60,000 training examples (NIST) database. For a small (U.S. Postal Service) database containing 7,000 training examples, the network with smaller capacity, LeNet 1, is better.

invariants that were used for generating the new examples. For example, for polynomial machines one can incorporate *a priori* information about invariance by using the convolution of an inner product in the form $(x^T A x^*)^d$, where x and x^* are input vectors and A is a symmetric positive definite matrix reflecting the invariants of the models.[18]

One can also incorporate another (geometrical) type of *a priori* information using only features (monomials) $x_i x_j x_k$ formed by pixels which are close each to other (this reflects our understanding of the geometry of the problem — important features are formed by pixels that are connected to each other, rather than pixels far from each other). This essentially reduces (by a factor of millions) the dimensionality of feature space.

Thus, although the theoretical foundations of Support Vector machines look more solid than those of Neural Networks, the practical advantages of the new type of learning machines still needs to be proved.[18a]

5.11 WISDOM OF STATISTICAL MODELS

In this chapter we introduced the Support Vector machines which realize the Structural Risk Minimization inductive principle by

(i) Mapping the input vector into a high-dimensional feature space using a nonlinear transformation.

(ii) Constructing in this space a structure on the set of linear decision rules according to the increasing norm of weights of canonical hyperplanes.

(iii) Choosing the best element of the structure and the best function within this element in order to minimize the bound on error probability.

The implementation of this scheme in the algorithms described in this chapter, however, contained one violation of the SRM principle. To define the structure on the set of linear functions we use the set of canonical hyperplanes constructed with respect to vectors x from the training data.

---

[18] B. Schölkopf considered an intermediate way: he constructed an SV machine, generated new examples by transforming the SV images (translating them in the four principal directions), and retrained on the support vectors and the new examples. For the U.S. Postal Service database, this improves the performance from 4.0% to 3.2%.

[18a] In connection with heuristics incorporated in Neural Networks let me recall the following remark by R. Feynman: "We must make it clear from the beginning that if a thing is not a science, it is not necessarily bad. For example, love is not science. So, if something is said not to be a science it does not mean that there is something wrong with it; it just means that it is not a science." *The Feynman Lectures on Physics*, Addison-Wesley, 3-1, 1975.

According to the the SRM principle, the structure has to be defined *a priori* before the training data appear.

The attempt to implement the SRM principle *in toto* brings us to a new statement of the learning problem which forms a new type of inference. For simplicity we consider this model for the pattern recognition problem.

Let the learning machine that implements a set of functions linear in feature space be given $\ell + k$ vectors

$$x_1, \ldots, x_{\ell+k} \tag{5.49}$$

drawn randomly and independently according to some distribution function.

Suppose now that these $\ell + k$ vectors are randomly divided into two subsets: the subset

$$x_1, \ldots, x_\ell$$

for which the string

$$y_1, \ldots, y_\ell, \quad y \in \{-1, +1\}$$

describing classification of these vectors is given (the training set), and the subset

$$x_{\ell+1}, \ldots, x_{\ell+k}$$

for which the classification string should be found by the machine (test set). The goal of the machine is to find the rule that gives the string with the minimal number of errors on the given test set.

In contrast to the model of function estimation considered in this book, this model looks for the rule that minimizes the number of errors on the *given test set* rather than for the rule minimizing the probability of error on the admissible test set. We call this problem the *estimation of the values of the function at given points*. For the problem of estimating the values of function at given points the SV machines will realize the SRM principle *in toto* if one defines the canonical hyperplanes with respect to all $\ell + k$ vectors (5.49). (One can consider the data (5.49) as *a priori* information. *A posteriori* information is any information about separating this set into two subsets.)

Estimating the values of a function at given points has both a solution and a method of solution, that differ from those based on estimating an unknown function.

Consider for example the five digit zip–code recognition problem.[19] The existing technology based on estimating functions suggests recognizing the five digits $x_1, \ldots, x_5$ of the zip-code independently: first one uses the rules constructed during the learning procedures to recognize digit x_1, then one uses the same rules to recognize digit x_2 and so on.

[19]For simplicity we do not consider the segmentation problem. We suppose that all five digits of a zip–code are segmented.

The technology of estimating the values of a function suggests to recognizing all five digits jointly: the recognition of one digit, say x_1, depends not only on the training data and vector x_1, but also on vectors $x_2, \ldots, x_5$. In this technology one uses the rules that are in a special way adapted to solving a given specific task. One can prove that this technology gives more accurate solutions.[20]

It should be noted that for the first time this new view of the learning problem was found due to attempts to justify a structure defined on the set of canonical hyperplanes for the SRM principle.

5.12 WHAT CAN ONE LEARN FROM DIGIT RECOGNITION EXPERIMENTS?

Three observations should be discussed in connection with the experiments described in this chapter:

(i) The structure constructed in the feature space reflects *real-life problems* well.

(ii) The quality of decision rules obtained does not strongly depend on the type of SV machine (polynomial machine, RBF machine, two-layer NN). It does, however, strongly depend on the accuracy of the VC dimension (capacity) control.

(iii) Different types of machines use the same elements of training data as support vectors.

5.12.1 Influence of the Type of Structures and Accuracy of Capacity Control

The classical approach to estimating multidimensional functional dependencies is based on the following belief:

Real-life problems are such that there exists a small number of "strong features," simple functions of which (say linear combinations) approximate well the unknown function. Therefore, it is necessary to carefully choose a low-dimensional feature space and then to use regular statistical techniques to construct an approximation.

[20] Note that the local learning approach described in Section 4.5 can be considered as an intermediate model between function estimation and estimation of the values of a function at points of interest. Recall that for a small (Postal Service) database the local learning approach gave significantly better results (3.3% error rate) than the best result based on entire function estimation approach (5.1% obtained by LeNet 1, and 4.0% obtained by the polynomial SV machine).

This approach stresses: be careful on the stage of feature selection (this is an informal operation) and then use routine statistical techniques.

The new technique is based on a different belief:

Real-life problems are such that there exist a large number of "weak features" whose "smart" linear combination approximates the unknown dependency well. Therefore, it is not very important what kind of "weak feature" one uses, it is more important to form "smart" linear combinations.

This approach stresses: choose any reasonable "weak feature space" (this is an informal operation), but be careful at the point of making "smart" linear combinations. From the perspective of SV machines, "smart" linear combinations corresponds to the capacity control method.

This belief in the structure of real-life problems has been expressed many times both by theoreticians and by experimenters.

In 1940, Church made a claim that is known as the Turing–Church Thesis[21]:

All (sufficiently complex) computers compute the same family of functions.

In our specific case we discuss the even stronger belief that linear functions in various feature spaces associated with different convolutions of the inner product, approximate the same set of functions if they possess the same capacity.

Church made his claim on the basis of pure theoretical analysis. However as soon as computer experiments became widespread, researchers were unexpectedly faced a situation that could be described in the spirit of Church's claim.

In the 1970s and in the 1980s a considerable amount of experimental research was conducted in solving various operator equations that formed ill-posed problems, in particular in density estimation. A common observation was that the choice of the type of regularizers $\Omega(f)$ in (4.32) (determining a type of structure) is not as important as choosing the correct regularization constant $\gamma(\delta)$ (determining capacity control).

In particular in density estimation using the Parzen window

$$p(x) = \frac{1}{\ell}\sum_{i=1}^{\ell}\frac{1}{\gamma^n}K\left(\frac{x-x_i}{\gamma}\right),$$

a common observation was: if the number of observations are not "very small", the type of kernel function $K(u)$ in the estimator is not as important

[21]Note that the thesis does not reflect some proved fact. It reflects the belief in the existence of some law that is hard to prove (or formulate in exact terms).

as the value of the constant γ. (Recall that the kernel $K(u)$ in Parzen's estimator is determined by the functional $\Omega(f)$, and γ is determined by the regularization constant.)

The same was observed in the regression estimation problem where one tries to use expansions in different series to estimate the regression function: if the number of observations is not "very small" the type of series used is not as important as the number of terms in the approximation. All these observations were done solving low-dimensional (mostly one-dimensional) problems.

In the described experiments we observed the same phenomena in very high-dimensional space.

5.12.2 SRM Principle and the Problem of Feature Construction

The "smart" linear combination of the large number of features used in the SV machine has an important common structure: the set of support vectors. We can describe this structure as follows: along with the set of weak features (weak feature space) there exists a set of complex features associated with support vectors. Let us denote this space

$$u = (K(x, x_1), \ldots, K(x, x_N)) \in U,$$

where

$$x_1, \ldots, x_N$$

are the support vectors. In the space of complex features U, we constructed a linear decision rule. Note that in the bound obtained in Theorem 5.2 the expectation of the number of complex features plays the role of the dimensionality of the problem. Therefore one can describe the difference between the support vector approach and the classical approach in the following way:

To perform the classical approach well requires the human selection (construction) of a relative small number of "smart features" while the support vector approach selects (constructs) a small number of "smart features' automatically.

Note that the SV machines construct the Optimal hyperplane in the space Z (space of weak features) but not in the space of complex features. It is easy, however, to find the coefficients that provide optimality for the hyperplane in the space U (after the complex features are chosen). Moreover one can construct in the U space a new SV machine (using the same training data). Therefore one can construct two (or several) layers SV machine. In

other words one can suggest multi-stage selection of "smart features". As we remarked in Section 4.10, the problem of feature selection is however quite delicate (recall the difference between constructing sparse algebraic polynomials and sparse trigonometric polynomials).

5.12.3. Is the Set of Support Vectors a Robust Characteristic of the Data?

In our experiments we observed an important phenomenon: different types of SV machines optimal in parameters use almost the same support vectors: there exists a small subset of the training data (in our experiments less than 3% to 5% of data) that for the problem of constructing the best decision rule is equivalent to the complete set of training data, and that this subset of the training data is almost the same for different types of optimal SV machines (polynomial machine with the best degree of polynomials, RBF machine with the best parameter γ, and NN machine with the best parameter b.)

The important question is whether this is true for a wide set of real-life problems. There exists indirect theoretical evidence that this is quite possible. One can show that if a majority vote scheme, based on various support vector machines, does not improve performance, then the percentage of common support vectors of these machines must be high.

It is too early to discuss the properties of SV machines: the analysis of these properties now just started.[22] Therefore I would like to finish these comments with the following remark.

---

[22]After this book had been completed, C. Burges demonstrated that one can approximate the obtained decision rule

$$f(x) = \text{sign}\left\{\sum_{i=1}^{N} \alpha_i K(x, x_i) + \alpha_0\right\}$$

by the much simpler decision rules

$$f^*(x) = \text{sign}\left\{\sum_{i=1}^{M} \beta_i K(x, T_i) + \beta_0\right\}, \qquad M << N,$$

using the, so-called, generalized support vectors $T_1, \ldots, T_M$ (a specially constructed set of vectors).

To obtain approximately the same performance for the digit recognition problem, described in Section 5.7, it was sufficient to use an approximation based on $M = 11$ generalized support vectors per classifier instead of $N = 270$ (initially obtained) support vectors per classifier.

This means that for Support Vector machines there exists a regular way to synthesize the decision rules possessing the optimal complexity.

The SV machine is a very suitable object for theoretical analysis. It unifies various conceptual models:

(i) The SRM model. (That is how the SV machine initially was obtained. Theorem 5.1.)

(ii) The Data Compression model. (The bound in Theorem 5.2 can be described in terms of the compression coefficient.)

(iii) A universal model for constructing complex features. (The convolution of the inner product in Hilbert space can be considered as a standard way for feature construction.)

(iv) A model of real-life data. (A small set of support vectors might be sufficient to characterize the whole training set for different machines.)

In a few years it will be clear if such unification of models reflects some intrinsic properties of learning mechanisms, or if it is the next cul-de-sac.

Vladimir N. Vapnik

The Nature of Statistical Learning Theory

With 33 Illustrations

Springer

Vladimir N. Vapnik
AT&T Bell Laboratories
101 Crawfords Corner Road
Holmdel, NJ 07733 USA

Library of Congress Cataloging-in-Publication Data
Vapnik, Vladimir Naumovich.
The nature of statistical learning theory / Vladimir N. Vapnik.
p. cm.
Includes bibliographical references and index.
ISBN 0-387-94559-8 (alk. paper)
1. Computational learning theory. 2. Reasoning. I. Title.
Q325.7.V37 1995
006.3'1'015195—dc20 95-24205

Printed on acid-free paper.

Production coordinated by Bill Imbornoni; manufacturing supervised by Joseph Quatela.
Photocomposed copy prepared from the author's LaTeX file.
Printed and bound by R.R. Donnelley and Sons, Harrisonburg, VA.
Printed in the United States of America.

9 8 7 6 5 4 3 2 1

ISBN 0-387-94559-8 Springer-Verlag New York Berlin Heidelberg